Content layer progressive coding of digital maps

Forchhammer, Søren; Jensen, Ole Riis

Published in:
Proc. IEEE Data Compression Conf.

Link to article, DOI:
10.1109/DCC.2000.838163

Publication date:
2000

Document Version
Publisher's PDF, also known as Version of record

Content Layer Progressive Coding of Digital Maps

Søren Forchhammer and Ole Riis Jensen
Dept. of Telecommunication, 371, Technical University of Denmark
e-mail: sf@tele.dtu.dk, Ole.Riis.Jensen@dnk.xerox.com

Abstract

A new lossless context based method is presented for content progressive coding of limited bits/pixel images, such as maps, company logos, etc., common on the WWW. Progressive encoding is achieved by separating the image into content layers based on color level or other predefined information. Information from already coded layers are used when coding subsequent layers. This approach is combined with efficient template based context bi-level coding, context collapsing methods for multi-level images and arithmetic coding. Relative pixel patterns are used to collapse contexts. The number of contexts are analyzed. The new methods outperform existing coding schemes coding digital maps and in addition provide progressive coding. Compared to the state-of-the-art PWC coder, the compressed size is reduced to 60-70 % on our layered test images.

1 Introduction

In this paper a lossless method is introduced for progressive or layered coding of images with a limited number of distinct color or grey scale values (referred to as levels). Examples of such images are typically seen on web pages on the Internet depicting icons, company logos, maps or palletized images. We have chosen to focus on coding of maps which is an important application area. Maps appear among other places on web pages as eg. for the Yellow Pages and tourist bureaus.
Images of this type are often limited to relatively few colors, and are usually computer generated and often coded with lossless GIF or lossy JPEG.
A number of efficient lossless coding schemes are based on context coding as JBIG for bi-level coding [1] and [2] for grey scale images. In context based coding, arithmetic coding is performed for each pixel, x_t based on probabilities $p(x_t|c)$ conditioned on the context c derived from the causal data x^t. In bi-level image coding the context, c is efficiently specified by a template. Aside from bi-level images the number of contexts grows rapidly with increasing template size. To reduce the number of contexts, the context may be obtained by a mapping of the causal template pixels. For natural images prediction and differencing followed by quantization is an efficient way to reduce the number of contexts[2], but linear prediction is not suited for graphic data such as maps. Recently, methods aimed at these limited bits/pixel images have been presented. PWC [3] and RAPP [4] both code whether the current pixel is equal to one of the causal 4-neighbors. PWC uses edge maps to reduce contexts. RAPP

[0]This work was partially funded by Tele Danmark.

GIF	TIFF (LZW)	JPEG-LS	Bit-plane		RAPP (M=5)	PWC (flip, collapse)
			Direct	Gray		
49248	70608	66778	33298	30426	26472	20600

Table 1: Lossless code length (bytes) for the map shown in Fig. 1 (composite image).

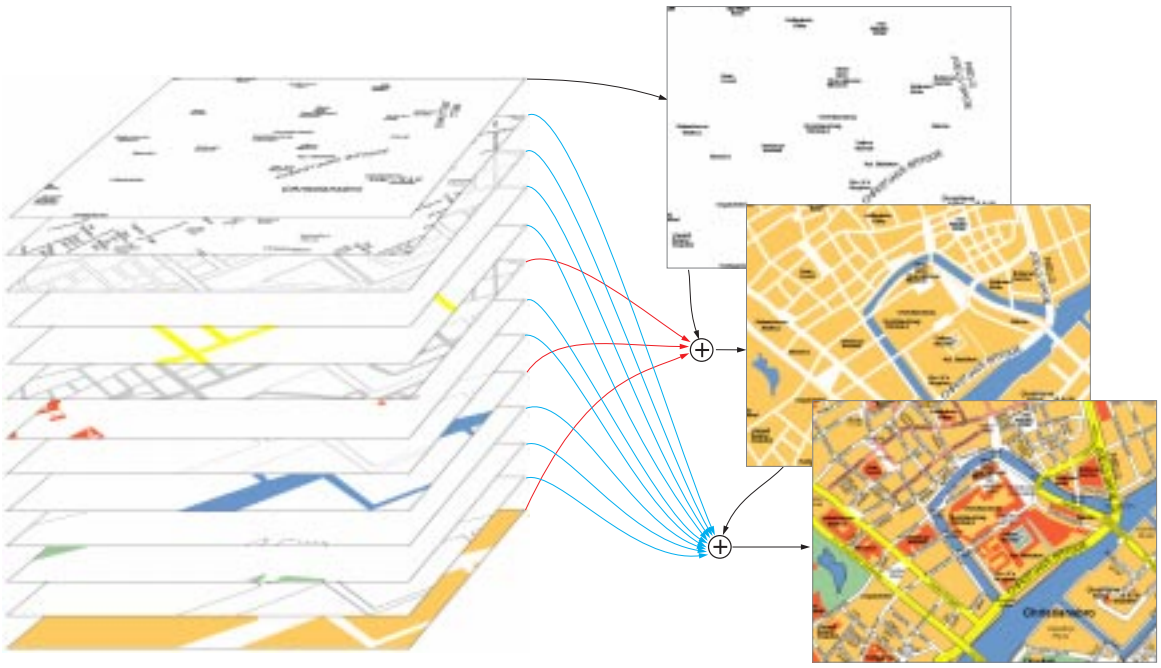

Figure 1: Test image of map (city center of Copenhagen). To the left, the layered image. To the right, progressive images leading to the composite image. Courtesy of GeoVision and Tele Danmark.

uses relative pixel patterns within the four neighbors. EIDAC [5] uses a bit-plane representation and selects bits from the current and previous bit-planes. It provides progression by bit-planes. In Table 1, results are given for some of the coding methods for the map of Fig. 1. PWC achieves the best result of these prior art methods.
Images are usually represented as *composite images* in a *single* layer of pixels, each represented by a number of bits needed to describe the color or grey tone level of the pixel. Images such as maps often originate from data which is generated as separate (bi-level) layers such as text, roads, buildings, etc., each of which may overlap other layers. A *layered image* consists of a number of layers each representing one level. The layers may be combined by some compostion rule to form a composite image. The layers may e.g. be stacked in a predefined order, such that image data of layers with higher priority visibly hide image data of lower priority layers. In this case, a composite image is obtained from a layered image by merging layers from the top layer to the bottom layer into a multi-level image with the same visual appearance (Fig. 1). We address the problem of efficiently coding layered images facilitating progression by layers. The RAPP algorithm was modified for residue coding in [6]. The approach is elaborated in this paper.
In Section 2 we introduce the layered image representation formally. Coding of both bi-level and multi-level layers are introduced. In Section 3 we analyze the number of contexts when using relative pixel patterns. Section 4 gives results for coding digital street maps.

2 Layered Coding

Let y_t denote a pixel of a composite image y^T with T pixels. Each pixel y_t takes on one of N values, { 1,...,N }, specifying the distinct color level. Let $\mathbf{x}_t$ denote a vector of binary pixels defining the layered image $\mathbf{x}^T$. $\mathbf{x}_t = (x_t(1), ..., x_t(N)), x_t(i) \in \{0, 1\}$. (In this definition all pixels are assumed to be defined in at least one layer avoiding the need for an additional background layer.) A composition rule specifies the (many-to-one) mapping from $\mathbf{x}^T$ to y^T. An example is a mapping specified by priority which may be expressed by

$$y_t = l_0, l_0 = min_l\{l|x_t(l) \neq 0\}. \tag{1}$$

A composite image with N distinct color levels may be split into a layered image by separating it into N non overlapping bi-level images each representing a distinct level from the composite image, i.e.

$$y_t = i \Rightarrow \begin{cases} x_t(j) = 1, & j = i \\ x_t(j) = 0, & j \neq i \end{cases} \tag{2}$$

where $x_t(j)$ are the split layers obtained from the composite image.
We introduce a *hybrid image representation*, as a mixture of the layered and the composite image. It may also be seen as a generalization. The hybrid image, $\mathbf{y}^T$ is represented by L image coding layers. Let $\mathbf{y}_t$ denote a vector pixel of the hybrid layered image $\mathbf{y}^T$. $\mathbf{y}_t = (y_t(1), ..., y_t(L))$, where the value of each $y_t(i)$ is taken from some subset of the N color levels. Image layer k has n_k distinct levels. Therefore, $\sum_{k=1}^{k=L} n_k = N$.
One example is obtained by coding $n_b < N$ bi-level image layers, together with one or more *residual (multi-level) image(s)* of the $N - n_b$ remaining levels. Residual layer i represents n_i levels so that each level is coded in exactly one coding layer.
By encoding each of the bi-level and residual images separately one by one (all in the usual raster-scan order), we obtain a *layered* or *progressive coding method*. As the layers are associated with specific contents of the image the coding is *content progressive*.

2.1 Context mappings

Efficient coding of a layered image is in our scheme achieved by by using the inter-layer depence by sharing information across the layers. One approach is to choose template pixels from already coded layers (eg. as was done for bit-planes in [5]), thus coding a layer relative to one or more other layers. Another approach we call SKIP pixel coding. In a particular layer, if a given pixel has already been coded in a layer of higher priority it does not need to be coded in the current or any of the lower layers. Such pixels are skipped by the encoder in our hybrid scheme and are thus denoted *skip pixels*. Skipped pixels will appear in the context of subsequent pixels. As the value is known at the decoder, we may readily use it. To reduce the number of contexts we may apply one of two mappings. The skip pixels may be labeled by a 'skip value' (s) in the subsequent contexts, thus collapsing the pixel values of all higher layers are into a single *skip value* in the current layer (i):

$$y_t < i \Rightarrow c(y_t) = s, \tag{3}$$

where $c(y_t)$ denotes the value y_t takes on as a context pixel.

The context alphabet becomes one larger than the alphabet (including background/'to-be-coded') of the pixels being coded in the current layer. We use this mapping for multi-level SKIP coding. This increase of alphabet size may be avoided by instead labeling skip pixels as one of the colors in the current coding layer, say the background color, N:

$$y_t < i \Rightarrow c(y_t) = N. \tag{4}$$

The background color may also be interpreted as 'to-be-coded' until the last residual. The skip pixel approach may be combined with both bi-level and multi-level coding. Skip pixel coding is especially of benefit when coding split layers, as we avoid coding (and confusing the statistics with) the 'holes' which levels of higher priority has left.

2.2 Coding bi-level layers

When coding the bi-level layers independently of other layers, we use a 16 pixel template as the largest template in JBIG2 [1], [7] or free tree coding [8]. To utilize dependencies between two bi-level layers a template with 9 pixels from a previous bi-level layer and 4 from the current layer. The templates are the same as for JBIG2 refinement coding, but here applied to code a new layer instead of refining the old. To utilize dependencies from multiple levels of higher priority, skip pixel coding is applied. A 10 pixel template is used for skip pixel coding (3) having a ternary context alphabet. This is denoted SKIPs. Context mapping by labeling skip pixels as background (4) maintains a bi-level context alphabet and therefore a 16 pixel template is used. This is the default bi-level SKIP setting.

2.3 Coding multi-level layers

For coding of multi-level layers, PWC [3] and RAPP [4] may be applied.
In RAPP (Runs of Adaptive Pixel Patterns) contexts are calculated substituting the pixel values of template specified by the 4 causal 8-neighbors by labels A and up to D giving a new label to each new color in the context calculation may readily be generalized. For templates larger than four pixels, labels E, F, etc. are introduced if necessary. We refer to this as relative pixel pattern or template adaptive pixel patterns (TAPP). (In TAPP we do not introduce runs as done in RAPP.) As in RAPP coding, the first step is to code if the pixel has the value of one of the context pixels, conditioned on the relative pixel pattern context. If not, an escape character is coded and then the pixel value.
When we introduce SKIP pixels in TAPP for multi-level residual coding, we use a skip pixel in the context. We also modify the coding and statistics update such that if a context for a pixel contains skip pixels, the predicted value is not the skip value. For both bi-level and multi-level coding we apply the binary arithmetic coding also used [8] in to occurrence counts as the entropy coding.
The approach presented may readily be generalized. Each level of a given coding layer has a priority and a color assigned. Each coding layer may be coded by itself or using skip pixel residual coding, where the priorities combined with the values of the previously coded image specifies which levels may be skipped at each position. For each pixel of a given coding layer only the values of levels with higher (or possibly the same) priority are considered. Thus a given color may be split into several layers.
While being presented as a multiple pass algorithm it may, with the same final compression result, be implemented in a one-pass (multiple context) algorithm. For each pixel the layered coding is applied until the pixel value is actually coded. Processing the image in this way results in a non-progressive method.

2.4 Predicting missing pixel levels

Information of the values of all levels, i.e. the full layered image $\mathbf{x}_t$ is missing given only the composite image or at the decoder before all the layers are received. We may try to predict the missing information from the information available. Having prioritized layers (1) the composite pixel y_t having value i specifies the value of the layers $x_t(j),\ j \leq i$, but leaves the values $x_t(j), j > i$ unknown. We may try to predict the values of these 'holes', e.g. based on an assumption (or prior knowledge) of piece-wise continuous regions. A simple experiment was conducted applying snakes [9]. For a split bi-level layer (2) all the unspecified pixels may be chosen as either value. Thus a layer is predicted from the split layer of a composite image and the position of pixels of higher priority. Eg. coding the (white) roads extracted from the composite map in Fig. 1 requires 6568 bytes using template coding. Filling the holes using snakes reduces the code length to 3788 bytes, which is almost as little as the code length (3216 bytes) having the original layer.
For a layered image, a layer (yet) not available may be predicted from those available. Eg. contours are common in maps. A contour layer may be predicted from the layer the contour applies to. A simple approach is to choose the pixels colored by a morphological dilation operation on the roads. This may be used at the decoder displaying the predicted contours.

3 Number of Contexts for Adaptive Pixel Patterns

An important reason for mapping contexts is to reduce the number of contexts. This section examines the number of contexts using relative pixel patterns as in TAPP and some variations thereof. Having a template of I pixels, we may encounter up to I different relative values. Let $C(I)$ denote the total number of contexts for I template pixels. Let $C_j(I)$ denote the number, out of the $C(I)$ contexts, having j different colors. Let N denote the size of the alphabet. The number of different contexts may recursively be calculated as follows: $C_j(I+1)$ is obtained by adding a new color to one of the $C_{j-1}(I)$ contexts having $j-1$ different colors or adding one of the j colors already present in the $C_j(I)$ contexts. The recursion is given by

$$C_j(I+1) = C_{j-1}(I) + jC_j(I), \quad 1 < j \leq N, j \leq I+1, \tag{5}$$

which is initialized by

$$C_1(I+1) = C_1(I) = 1, \tag{6}$$

$$C_j(I) = 0, j > I \vee j > N, \tag{7}$$

which is consistent with $C_I(I) = 1$.
The total number of contexts for I template pixels may thereafter be calculated by

$$C(I) = \sum_{j=1}^{I'} C_j(I). \tag{8}$$

where $I' = \min\{I, N\}$.
Consider the situation where we code one of the j colors within the template or an escape character if the color was not in the template. In this case, the number of free parameters, $P_j(I)$ for the class of contexts specified by I and j is given by

$$P_j(I) = \sum_{j=1}^{I} jC_j(I), \quad N > I. \tag{9}$$

I	1	2	3	4	5	6	7	8	9	10
$C(I)$	1	2	5	15	52	203	877	4140	21.147	115.975
$P(I)$	1	3	10	37	151	674	3263	17.007	94.828	562.785

Table 2: The number of possible contexts $C(I)$ and free parameters $P(I)$ for template size I using relative patterns ($N > I$).

Summing over j we may calculate the total number of free parameters $P(I)$ for the I pixel template. Using the recursions (5) we calculate the number of contexts. The results are given in Table 2. With respect to the number of contexts it is quite feasible to work with 9-10 template pixels independently of the alphabet size. It is possible to combine the relative pixel patterns with the the actual color up to say $f < I$ colors. For a given number j of different colors the number of contexts become

$$C_C(I) = \sum_{j=1}^{f} \frac{N!}{(N-j)!} C_j(I) + \sum_{j=f+1}^{I} \frac{N!}{(N-f)!} C_j(i). \tag{10}$$

Algorithm Context could be used to prune the context tree. A simple alternative is given below. These could also be used to choose the set, algorithm Context will examine. A way to keep the number of contexts down while allowing for large templates when only few colors are present locally is to impose a maximum M on the number of colors in the template. If template pixel $k+1 \leq I$ has a new color leading to $M+1$ template colors, the template is reduced to k pixels. This leads to the following expression on the number of contexts:

$$C_M(I) = \sum_{j=1}^{M} C_j(I) + \sum_{k=M}^{I-1} C_M(k), \qquad M < I. \tag{11}$$

4 Results

Our main interest is efficient progressive coding of street map data. The algorithms were tested on a number of street maps. The layered and composite map of Fig. 1 is our main data set having 723 by 546 pixels, but the algorithms were also tested on other maps.

The first results presented here concern the map shown in Fig. 1, represented both as a layered image $\mathbf{x}^T$ with 12 overlapping layers, and as its composite equivalent y^T. Table 3 shows results for the image coded as the original 12 overlapping layers. Table 4 gives the results coding the 12 non-overlapping layers extracted from the composite equivalent (2). Table 5 gives the results for our content progressive scheme using hybrid representations of the image.

Table 1 showed that the PWC [3] and RAPP [4] algorithms outperform prior methods on the composite image, and these two algorithms are able to obtain a reduction in compression size of 45-55% compared to the GIF file size. From Table 3 we see that by having access to the original layered bi-level data and coding the layers with a template coder a further reduction may be achieved. The best result using two layer templates for some layers is almost 3 times better than GIF. It should be noted that we actually have more information in the full layered representation than in the composite representation.

From Tables 3 and 4 we can extract and calculate the best results depending on the (desired) attributes of the input and output. The best result for the layered image,

$\mathbf{x}^T$ is 14.878 bytes (Table 3). For five of the layers, context pixels were chosen from other layers, introducing a dependence. If we want to be able to choose any ordering of the layers, all layers should be coded independently. The best result in this case is 17.276 bytes obtained by the free tree [8] and coding the text layer as the two coding layers it was split into in Fig. 1. Having the layered image at the time of en-coding but only being required to represent the composite image, the best result is 14.434 bytes. (Obtained by the 'Best' result from Table 3 combined with the best result from Table 4 for layers 8 and 10.) When we only need to represent the composite image we do not need to code the last layer. When the encoder only has access to the composite version, the best result is 18.133 bytes using the free tree for layers 0 and 1 and residue SKIP (9 pixel template) coding for the rest. Using a 16 pixel template instead of the free tree the result is 19.629 bytes.

For the set-ups above, the best results are all better than the PWC result. The overall best result reduced the code length to 70 % of PWC. Having only the composite image at the encoder the code length was reduced to 88 %. When the layered image is available the code actually carries more information ($\mathbf{x}^T$ or $\mathbf{y}^T$) than just the composite image (y^T) which obviously has lower (or in special cases equal) entropy.

To obtain content progressive encoding we must decide which layers of the image to present first to the decoder. On a map, the layers that first enables a client to identify a location is probably the text and the roads. In Fig. 1 we have shown a progressive version with (a subset of) the text layer, the roads and the water. Table 5 shows that our content progressive scheme, using a 16 pixel template coder for bi-level layers and SKIP coding (9 pixel template) for the residual image, improves the code length of RAPP and PWC, while also providing content progressive coding. Using the residual coder, we can have the progression over the desired layers and then code the residual in one coding layer. The loss introduced by performing progression by levels derived from composite image is also quite small. Thus, this combination gives an effective selection of contents. Comparing the first and the last column, the cost of coding the hybrid image without knowledge of the original overlapping layers is in this case only 1496 bytes.

Initial tests of the algorithms were also conducted on a layered digital map from which a composite map for printing as a street map may be extracted. This map was provided by KRAK, Denmark. On the composite image, PWC performed the best (Table 7). Coding the layered image as bi-level layers using the free tree reduced the code length to 72 %. Hybrid codings, coding a residual layer after coding the text layer by a free tree and JBIG-2 yielded a code length of 249014 and 263821 bytes, respectively. The map has 27 color levels but it was actually provided as $\mathbf{y}^T$ in the six themes listed in Table 8 and 9. The themes were split to obtain a full bi-level layered description ($\mathbf{x}^T$). Coding the image theme by theme improves the performance of TAPP and PWC (Table 8). PWC just being slightly better now. The results of coding the bi-level layers are given in Table 9 accumulated by theme. The SKIP coding codes the composite image by split layers yielding a 7 % shorter code length. Picking the best result of the free tree and SKIP coding at a bi-level basis, yielded a hybrid code length of 142.640 bytes, 42 % less than PWC and more than 7 times less than GIF on the composite image (Table 7). These results suggests that again we can achieve content layer progressive coding maintaining or even improving the performance of the state of the art PWC coder.

Finally tests were carried out on maps from the test set of [4]. These maps were only available in the ordinary composite version. The initial tests (Table 10) show that efficient progressive coding may be obtained by skip coding of bi-level layers and in the residual layers extracted from the composite image. The progressive SKIP coding yields slightly longer code lengths compared with the PWC in this test.

5 Conclusion

The presented content progressive lossless coding scheme achieves very good overall compression results for limited bits/pixel images such as maps. The compression factor was improved up to a factor of 7 compared to GIF. The code length from layered image data was reduced to as little as 60-70 % in comparison with PWC. At the same time progression was provided. Many images may be split into content layers, such as text, buildings, etc., and we code these layers separately. For efficient coding, information from already coded layers are used to determine pixels in higher layers which may be skipped (skip pixels) in this and all lower layers. The principle of skip pixel coding introduced is combined with existing efficient template based context bi-level coding [7], context collapsing methods for multi-level images [4] and arithmetic coding. It is also possible to extract layers from ordinary composite images to provide efficient progressive coding. All in all, the new scheme may thus be used as a flexible tool for efficient progressive coding of digital maps.

References

[1] ISO/IEC 14492 CD, "Coded Representension of Picture and Audio Information - Lossy/Lossless coding of bi-level images (JBIG2)", *ISO/IEC JTC1/SC29/WG1*

[2] M. J. Weinberger, J. Rissanen, and R. B. Arps, "Applications of Universal Context Modelling to Lossless Compression of Grey-Scale Images", *IEEE Trans. Image Processing*, vol. 5, no. 4, April 1996, pp. 575-586.

[3] P. J. Ausbeck Jr., "A Streaming Piecewice-Constant Model", *Proceedings Data Compression Conference*, March 1999, pp. 208-217.

[4] Viresh Ratnaker, "RAPP: Lossless Image Compression with Runs of Adaptive Pixel Patterns", *Thirty-Second Asilomar Conference on Signals, Systems and Computers*, November 1998.

[5] Y. Yoo, Y. Kwon, and A. Ortega, "Embedded Image-Domain Adaptive Compression of Simple Images", *Thirty-Second Asilomar Conference on Signals, Systems and Computers*, November 1998.

[6] O. R. Jensen and S. Forchhammer, "Content Progressive Coding of Limited Bits/Pixel Images", *Proc. IEEE 3rd Workshop Multimedia Signal Process.*, Sept. 1999, pp. 419-424.

[7] P. G. Howard, F. Kossentini, B. Martins, S. Forchhammer, and W. J. Rucklidge, "The Emerging JBIG2 Standard", *IEEE trans. Circuits and Systems for Video Tech.*, vol. 8, no. 7, November 1998, pp 838-848.

[8] B. Martins and S. Forchhammer, "Tree Coding of Bilevel Images", *IEEE Trans. Image Processing*, vol. 7, no. 4, April 1998, pp. 517-528.

[9] M. Kass, A. Witkin, and D. Terzopolous, "Snakes: Active Contour Models", *Journal of Computer Vision*, 1988, pp 321-331.

Layer description	overlapping layers					
	JBIG1	Template	Temp+ref	Free tree	Best	RAPP
0, Text	9514	8952	8952	8772	* 8280	11229
1, Road contours	5103	4512	4512	3556	3556	8005
2, Roads, pink	382	340	(r4) 291	248	248	649
3, Roads, yellow	656	616	(r4) 287	456	(r4) 287	961
4, Roads, white	3587	3216	(r1) 447	2140	(r1) 447	5180
5, Buildings	966	924	924	792	792	1341
6, Water, cont.	522	492	492	416	416	1057
7, Water	510	480	(r6) 124	392	(r6) 124	649
8, Parks, cont.	188	176	176	156	156	329
9, Parks	189	172	(r8) 76	132	(r8) 76	257
10, Land, cont.	735	516	516	388	388	909
11, Land	409	408	(r10) 108	320	(r10) 108	569
Total	22761	20804	16905	17768	14878	31135
Composite	22352	20396	16797	17448	14770	30566

Table 3: Layered image. Code lengths (bytes) for the layers of the Copenhagen map. (rl) means that this layer is coded relative to layer l. * the result is obtained by free tree coding on the text split in two layers.

Layer description	non overlap. layers				
	Template	RAPP	SKIPs	SKIP	Free Tree
0, Text	8952	11229	9470	8952	8772
1, Road contours	6008	8857	6286	5488	5184
2, Roads, pink	808	1161	834	520	720
3, Roads, yellow	1679	2245	870	728	1500
4, Roads, white	6568	8321	1326	2680	6028
5, Buildings	1683	2325	1478	1240	1572
6, Water, cont.	488	753	758	428	452
7, Water	896	1093	558	328	812
8, Parks, cont.	36	65	386	32	24
9, Parks	292	389	450	120	260
10, Land, cont.	272	361	498	184	220
11, Land	6488	8573	322	1212	5944
Total	34170	45372	23252	21912	31488
Composite	27682	36799	22930	20700	25544

Table 4: Composite image. Code lengths (bytes) for split layers extracted from the map . SKIPs uses an explicit skip symbol in the contexts.

Coding method	Layer description	Size	Size	Size	Size
Template	Text	8952	8952	8952	8952
Template	Road contours	4512		4512	6008
Template	Roads, white		3216	(r1) 447	
Template	Roads, pink			340	
Template	Roads, yellow			616	
Skip coding (9 pix.tem.)	Residual	4669	10912	3016	4669
Skip coding (M=4)	Residual	(7825)	(12636)	(4854)	(7825)
Total		18133	23080	17883	19629

Table 5: Hybrid coding. Content progressive code lengths (bytes) for the map in different progressive versions (hybrid image). Last column is for split layers.

Template size	4	5	6	7	8	9
Code length	28140	26533	25457	22835	21973	21002

Table 6: Code lengths for the Copenhagen map for different template sizes using template adaptive pixel patterns (TAPP).

GIF	TIFF (LZW)	PNG	PWC (collapse)	RAPP (M=8)	TAPP 9 pix	JBIG-2 16 pix	Free Tree
1015170	1376232	687673	254243	320810	268098	297612	182636

Table 7: KRAK image, map of Lyngby, Denmark. 3939 x 2760 pixels, 27 layers. JBIG-2 and Free Tree coding was applied to the individual layers.

Layer theme description	Coding Method			
	GIF	TIFF	PWC	TAPP
Symbols (7)	36907	362034	4007	4062
Text (1)	225095	474814	84506	87254
Grid (1)	60967	236340	1355	430
Lines (4)	48406	371132	6252	7022
Roads (7)	397342	959444	86140	79722
Areas (7)	283047	810044	64138	68930
Total	1051764	3213808	246398	247420

Table 8: KRAK image. Code lengths for coding by themes. (n) refers to the number of bi-level layers in the theme.

Layer theme description	Coding Method			
	Template	Free Tree	SKIP	Best
Symbols (7)	6678	4836	4398	3108
Text (1)	75363	60556	81602	60556
Grid (1)	220	76	1222	76
Lines (4)	6542	4308	6444	3658
Roads (7)	104374	55036	85122	46540
Areas (7)	104435	57824	50330	28702
Total	297612	182636	229118	142640

Table 9: KRAK image. Code lengths for bi-level layers accumulated by themes. (n) refers to the number of bi-level layers in the theme.

Image	N	RAPP	TAPP	SKIP	PWC
2b09cf-	79	25288	19256	17987 (9)	18096
2b09cq-	75	27232	20380	19348 (5)	18126
2b0avt-	78	18472	14048	13380 (7)	13083

Table 10: Code lengths for 560x560 Japanese street maps [4]. (X) refers to the number of bilevel layers of the skip coding.